How to Prove Undecidability or Non-Turing-Recognizability in This Course

To prove that a given language is undecidable:

- Construct a (mapping) reduction from another language already known to be undecidable to the given language.
- This known undecidable language can be any language for which undecidability has been proved in the textbook, in lectures, in class handouts, or in homework problems (but you should cite the appropriate reference).
- How do you decide which existing language to use to reduce to the given language? You may be given a hint (or told outright), or you may be expected to figure it out for yourself. An obvious choice to consider is $A_{\rm TM}$ or its complement.
- Prove that your reduction has the desired properties (i.e., that it truly is a reduction from that undecidable language to the given language).

To prove that a given language is non-Turing-recognizable:

Either do both of these:

- Prove that its complement is Turing-recognizable.
- Prove that its complement is undecidable.

Or:

- Construct a (mapping) reduction from another language already known to be non-Turing-recognizable to the given language.
- This known non-Turing-recognizable language can be any language for which non-Turing-recognizability has been proved in the textbook, in lectures, in class handouts, or in homework problems (but you should cite the appropriate reference).
- How do you decide which existing language to use to reduce to the given language? You may be given a hint (or told outright), or you may be expected to figure it out for yourself. An obvious choice to consider is $\overline{A_{\rm TM}}$.
- Prove that your reduction has the desired properties (i.e., that it truly is a reduction from that non-Turing-recognizable language to the given language).